ADAPTIVE FILTERING

- Configurations and applications

- Steepest descent method

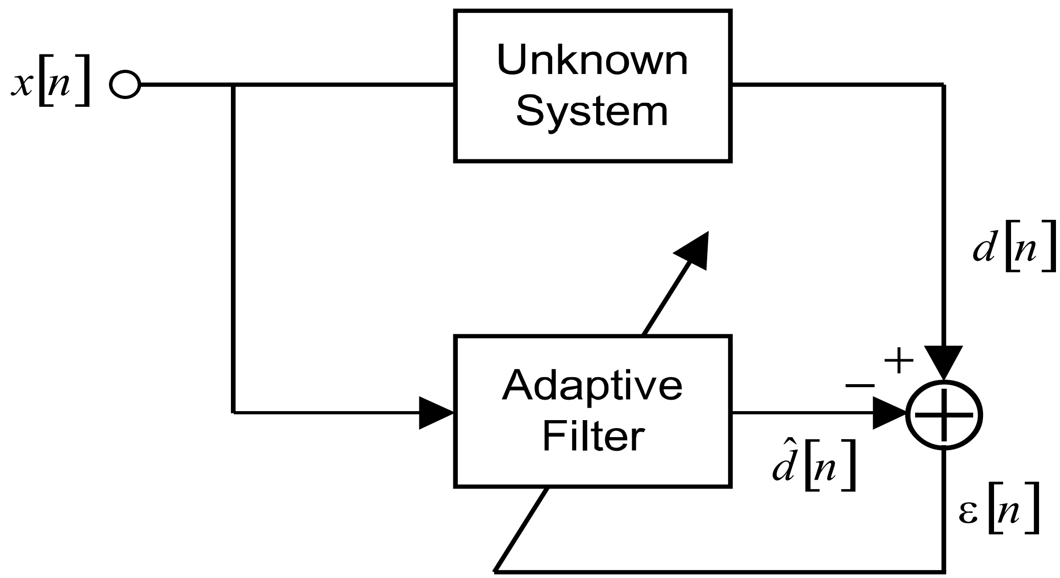
- Least mean squares (LMS)

- Recursive least squares (RLS)

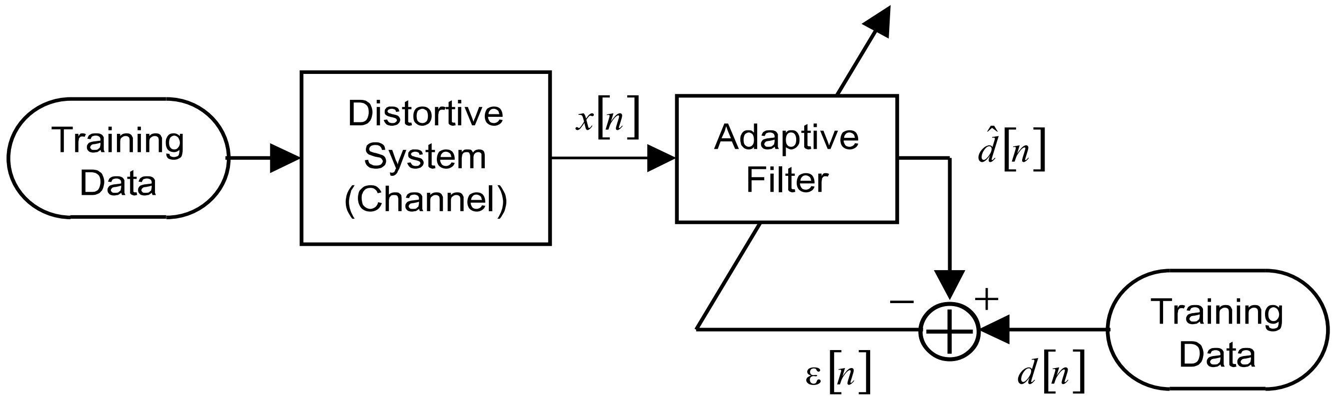
ADAPTIVE FILTER CONFIGURATIONS AND APPLICATIONS

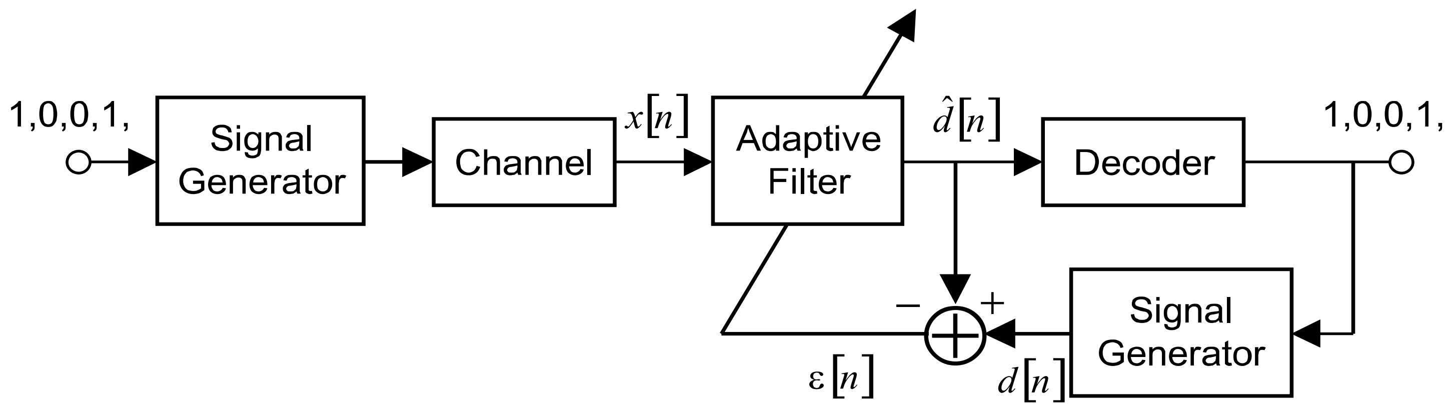
- System identification
- System equalization
- Linear prediction
- Noise cancelation

SYSTEM IDENTIFICATION

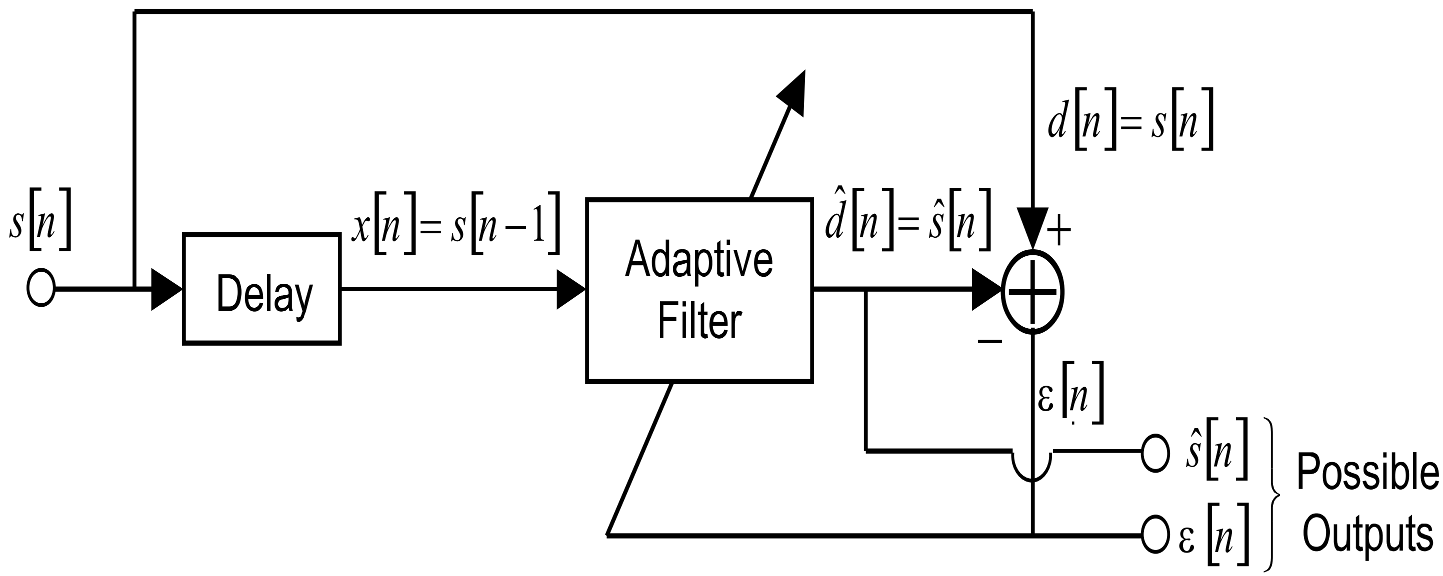


SYSTEM EQUALIZATION





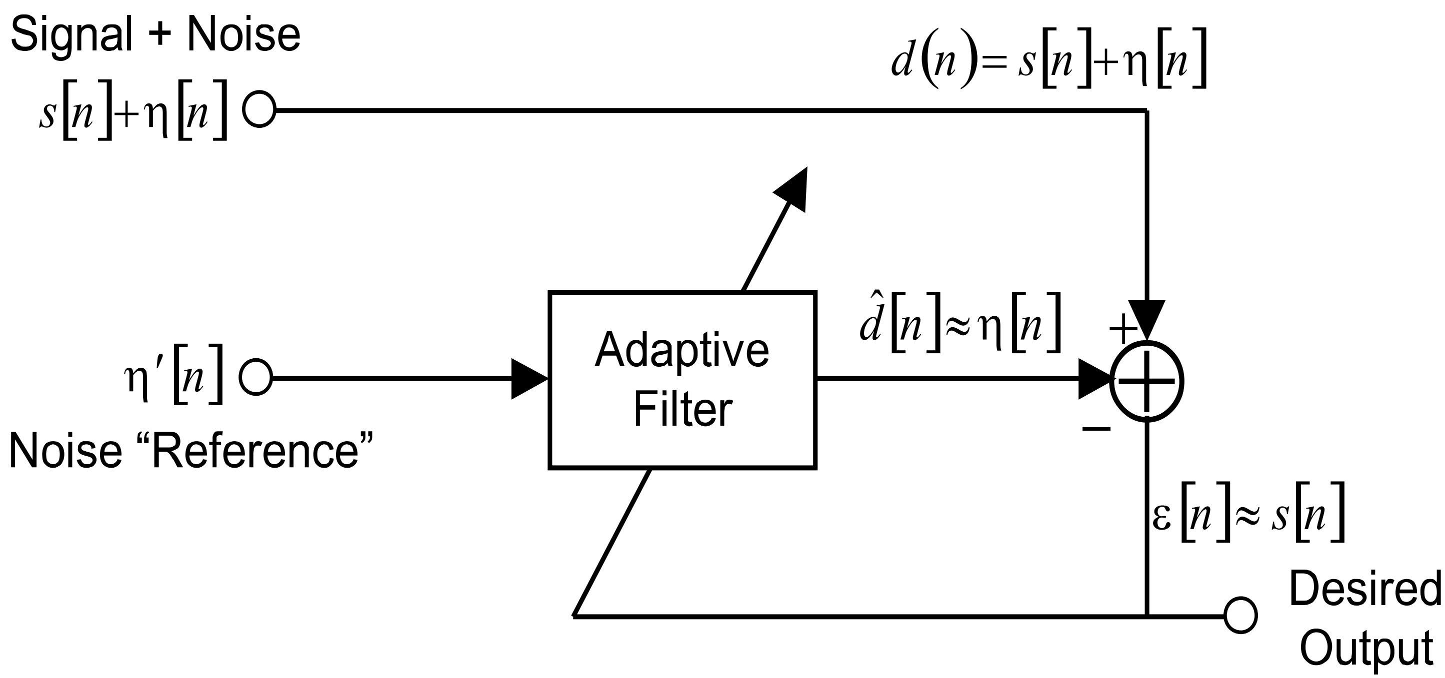
LINEAR PREDICTION



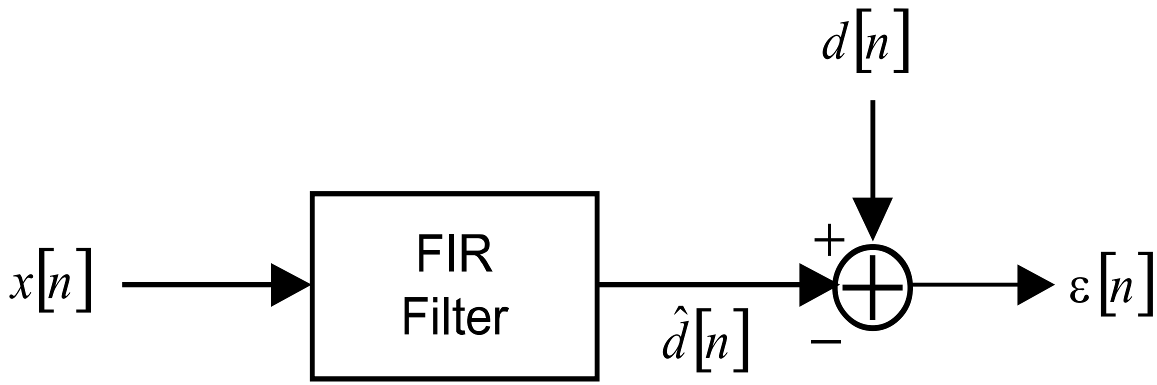
Adaptive Line Enhancer: output is $\hat{s}[n]$

Linear Predictive Coding: output is $\varepsilon[n]$

NOISE (INTERFERENCE) CONCELATION



WIENER FILTER REVIEW



$$h[n] = \begin{cases} w_n & 0 \le n \le P-1 \\ 0 & \text{otherwise} \end{cases}$$

$$\widehat{d}[n] = \sum_{k=0}^{P-1} w_k x[n-k] = \mathbf{w}^T \tilde{\boldsymbol{x}}[n]$$

WIENER FILTERING EQUATIONS

WIENER-HOPF EQUATION

$$\boxed{\mathbf{R}_{\boldsymbol{x}}\mathbf{w}_{\mathsf{o}} = \tilde{\mathbf{r}}_{d\boldsymbol{x}}}$$

$$\mathbf{R}_{\boldsymbol{x}} = \mathcal{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^{*T}[n]\right\} \qquad \mathbf{r}_{d\boldsymbol{x}} = \mathcal{E}\left\{d[n]\boldsymbol{x}^{*}[n]\right\}$$

MEAN-SQUARE ERROR

$$\sigma^2_{\varepsilon_{\mathsf{o}}} = R_d[0] - \mathbf{w}_{\mathsf{o}}^{*T}\tilde{\mathbf{r}}_{d\boldsymbol{x}} \qquad \varepsilon_{\mathsf{o}}[n] = d[n] - \mathbf{w}_{\mathsf{o}}^{T}\tilde{\boldsymbol{x}}[n]$$

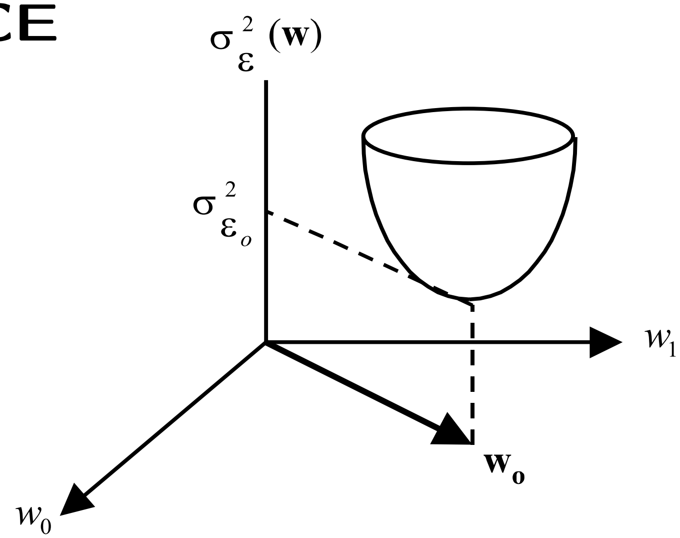
SOLUTION BY ITERATION

ERROR EQUATION

$$\sigma_\varepsilon^2(\mathbf{w}) = \mathcal{E}\left\{|\varepsilon[n]|^2\right\} = \mathcal{E}\left\{(d[n] - \mathbf{w}^T\tilde{\boldsymbol{x}}[n])(d[n] - \mathbf{w}^T\tilde{\boldsymbol{x}}[n])^*\right\}$$

$$\Longrightarrow \qquad \sigma_\varepsilon^2(\mathbf{w}) = R_d[0] - \mathbf{w}^{*T}\tilde{\mathbf{r}}_{d\boldsymbol{x}} - \tilde{\mathbf{r}}_{d\boldsymbol{x}}^{*T}\mathbf{w} + \mathbf{w}^{*T}\mathbf{R}_{\boldsymbol{x}}\mathbf{w}$$

ERROR SURFACE



SOLUTION BY ITERATION (cont'd.)

ERROR EQUATION

$$\sigma_\varepsilon^2(\mathbf{w}) = R_d[0] - \mathbf{w}^{*T}\tilde{\mathbf{r}}_{d\boldsymbol{x}} - \tilde{\mathbf{r}}_{d\boldsymbol{x}}^{*T}\mathbf{w} + \mathbf{w}^{*T}\mathbf{R}_{\boldsymbol{x}}\mathbf{w}$$

METHOD OF STEEPEST DESCENT

$$\nabla_{\mathbf{w}^*}\sigma_\varepsilon^2 = -\tilde{\mathbf{r}}_{d\boldsymbol{x}} + \mathbf{R}_{\boldsymbol{x}}\mathbf{w}$$

$$\mathbf{w}[n+1] = \mathbf{w}[n] - \mu\nabla_{\mathbf{w}^*}\sigma_\varepsilon^2$$

$$\boxed{\mathbf{w}[n+1] = \mathbf{w}[n] + \mu(\tilde{\mathbf{r}}_{d\boldsymbol{x}} - \mathbf{R}_{\boldsymbol{x}}\mathbf{w}[n])}$$

SOLUTION BY ITERATION (cont'd.) - AN ALTERNATIVE EXPRESSION

MEAN-SQUARE ERROR

$$\sigma_\varepsilon^2(\mathbf{w}) = \mathcal{E}\left\{|\varepsilon[n]|^2\right\} = \mathcal{E}\left\{\varepsilon[n](d[n] - \mathbf{w}^T\tilde{\boldsymbol{x}}[n])^*\right\}$$

GRADIENT $\quad \nabla_{\mathbf{w}^*}\sigma_\varepsilon^2 = -\mathcal{E}\left\{\varepsilon[n]\tilde{\boldsymbol{x}}^*[n]\right\}$

EQUATION OF STEEPEST DESCENT

$$\boxed{\mathbf{w}[n+1] = \mathbf{w}[n] + \mu\mathcal{E}\left\{\varepsilon[n]\tilde{\boldsymbol{x}}^*[n]\right\}}$$

PERFORMANCE ANALYSIS

Rewrite equation:

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu(\tilde{\mathbf{r}}_{d\boldsymbol{x}} - \mathbf{R}_{\boldsymbol{x}}\mathbf{w}[n])$$

as

$$\mathbf{u}[n+1] = (\mathbf{I} - \mu\mathbf{R}_{\boldsymbol{x}})\mathbf{u}[n] \qquad \text{where} \quad \mathbf{u}[n] = \mathbf{w}[n] - \mathbf{w}_{\mathrm{o}}$$

This implies

$$\mathbf{u}[n] = (\mathbf{I} - \mu\mathbf{R}_{\boldsymbol{x}})^n\mathbf{u}[0]$$

SOME ALGEBRAIC STEPS

$$
\begin{aligned}
\mathbf{R}_{\boldsymbol{x}} &= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{*T} \\
&= \begin{bmatrix} | & | & & | \\ \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_P \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_P \end{bmatrix} \begin{bmatrix} -- & \mathbf{e}_1^{*T} & -- \\ -- & \mathbf{e}_2^{*T} & -- \\ & \vdots & \\ -- & \mathbf{e}_P^{*T} & -- \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
(\mathbf{I} - \mu \mathbf{R}_{\boldsymbol{x}})^n &= \mathbf{E}(\mathbf{I} - \mu \mathbf{\Lambda})^n \mathbf{E}^{*T} \\
&= \mathbf{E} \begin{bmatrix} (1-\mu\lambda_1)^n & & & 0 \\ & (1-\mu\lambda_2)^n & & \\ & & \ddots & \\ 0 & & & (1-\mu\lambda_P)^n \end{bmatrix} \mathbf{E}^{*T}
\end{aligned}
$$

CONDITION FOR CONVERGENCE

$$
\begin{aligned}
\mathbf{u}[n] &= (\mathbf{I} - \mu \mathbf{R}_{\boldsymbol{x}})^n \mathbf{u}[0] = \mathbf{E}(\mathbf{I} - \mu \mathbf{\Lambda})^n \mathbf{E}^{*T} \mathbf{u}[0] \\
&= \mathbf{E} \begin{bmatrix} (1-\mu\lambda_1)^n & & & 0 \\ & (1-\mu\lambda_2)^n & & \\ & & \ddots & \\ 0 & & & (1-\mu\lambda_P)^n \end{bmatrix} \mathbf{E}^{*T} \mathbf{u}[0]
\end{aligned}
$$

This converges if and only if $|1 - \mu\lambda_i| < 1$ for all i, or

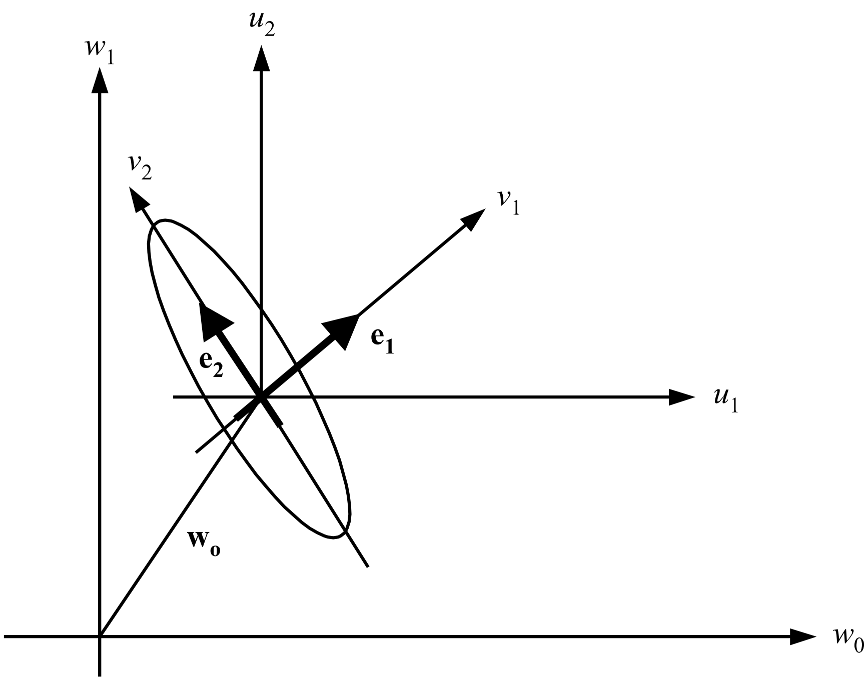
$$
\boxed{0 < \mu < \frac{2}{\lambda_{\text{max}}}}
$$

(λ_{max} is the *largest* eigenvalue of $\mathbf{R}_{\boldsymbol{x}}$)

MODES OF CONVERGENCE

"Modes" $v_i[n]$ are represented by a coordinant transformation:

$$\mathbf{v}[n] = \mathbf{E}^{*T}\mathbf{u}[n] = \mathbf{E}^{*T}(\mathbf{w}[n] - \mathbf{w}_\mathrm{o})$$



$$\begin{bmatrix} v_1[n] \\ v_2[n] \\ \vdots \\ v_P[n] \end{bmatrix} = \begin{bmatrix} -- & \mathbf{e}_1^{*T} & -- \\ -- & \mathbf{e}_2^{*T} & -- \\ & \vdots & \\ -- & \mathbf{e}_P^{*T} & -- \end{bmatrix} \mathbf{u}[n]$$

MODAL EXPRESSION FOR WEIGHTS

Time dependence of i^{th} mode: $v_i[n] = (1 - \mu\lambda_i)^n v_i[0]$

Combine this with the following relations:

$$\mathbf{w}[n] = \mathbf{w}_\text{o} + \mathbf{u}[n]$$

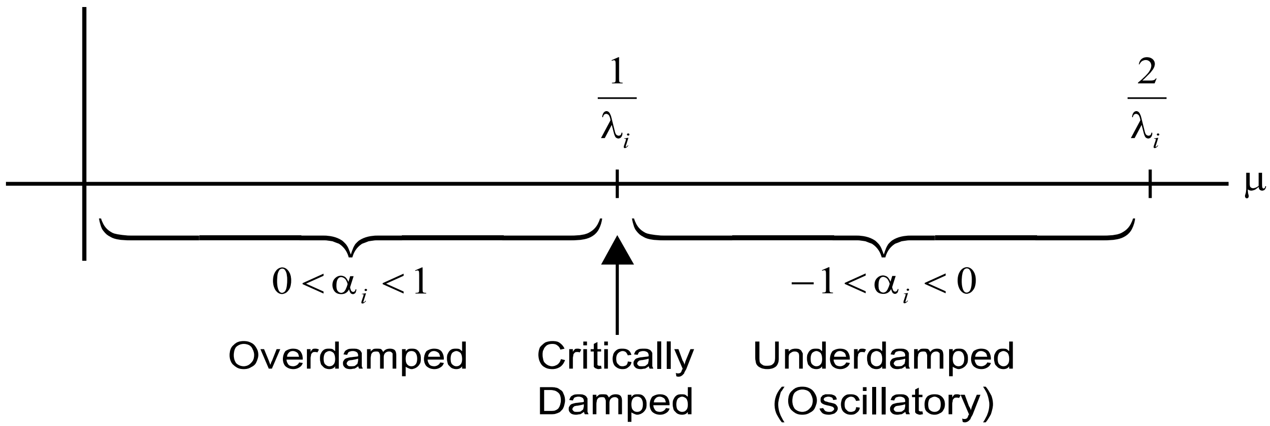
$$\mathbf{u}[n] = \mathbf{E}\mathbf{v}[n] = \begin{bmatrix} | \\ \mathbf{e}_1 \\ | \end{bmatrix} v_1[n] + \begin{bmatrix} | \\ \mathbf{e}_2 \\ | \end{bmatrix} v_2[n] + \cdots + \begin{bmatrix} | \\ \mathbf{e}_P \\ | \end{bmatrix} v_P[n]$$

to obtain:

$$\mathbf{w}[n] = \mathbf{w}_\text{o} + \sum_{i=1}^{P} \mathbf{e}_i v_i[0](1 - \mu\lambda_i)^n$$

CONVERGENCE TIME FOR MODES

REGIONS OF CONVERGENCE



$$\alpha_i = 1 - \mu\lambda_i$$

Time constant τ_i of the i^{th} mode is defined by

$$|1 - \mu\lambda_i|^{\tau_i} = e^{-1} \qquad \Longrightarrow \qquad \tau_i = \frac{1}{-\ln|1 - \mu\lambda_i|}$$

CONVERGENCE TIME FOR WEIGHTS

LARGEST TIME CONSTANT

$$\tau_{\mathbf{w}} = \frac{1}{-\ln(1-\mu\lambda_{\mathsf{min}})} \leq \frac{1}{\mu\lambda_{\mathsf{min}}}$$

Write μ as

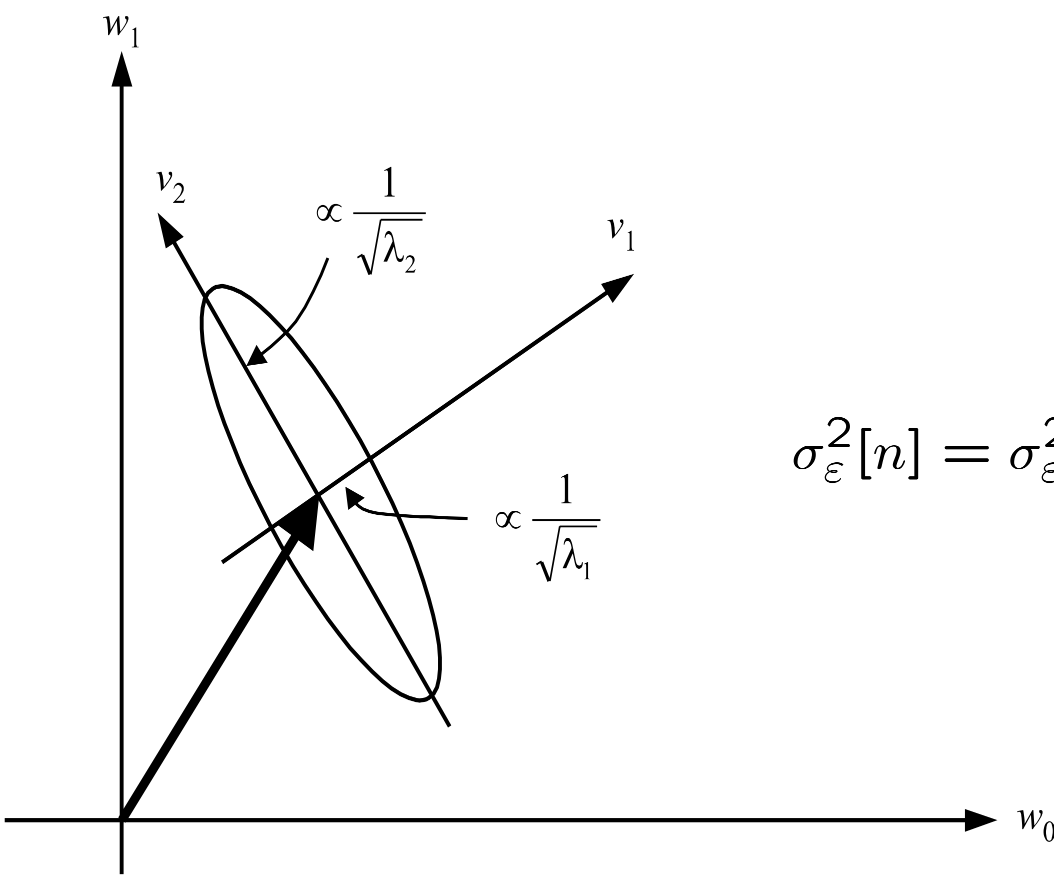
$$\mu = \rho\frac{2}{\lambda_{\mathsf{max}}} \qquad 0 < \rho < 1$$

then

$$\boxed{\tau_{\mathbf{w}} \leq \frac{1}{2\rho}\left(\frac{\lambda_{\mathsf{max}}}{\lambda_{\mathsf{min}}}\right)}$$

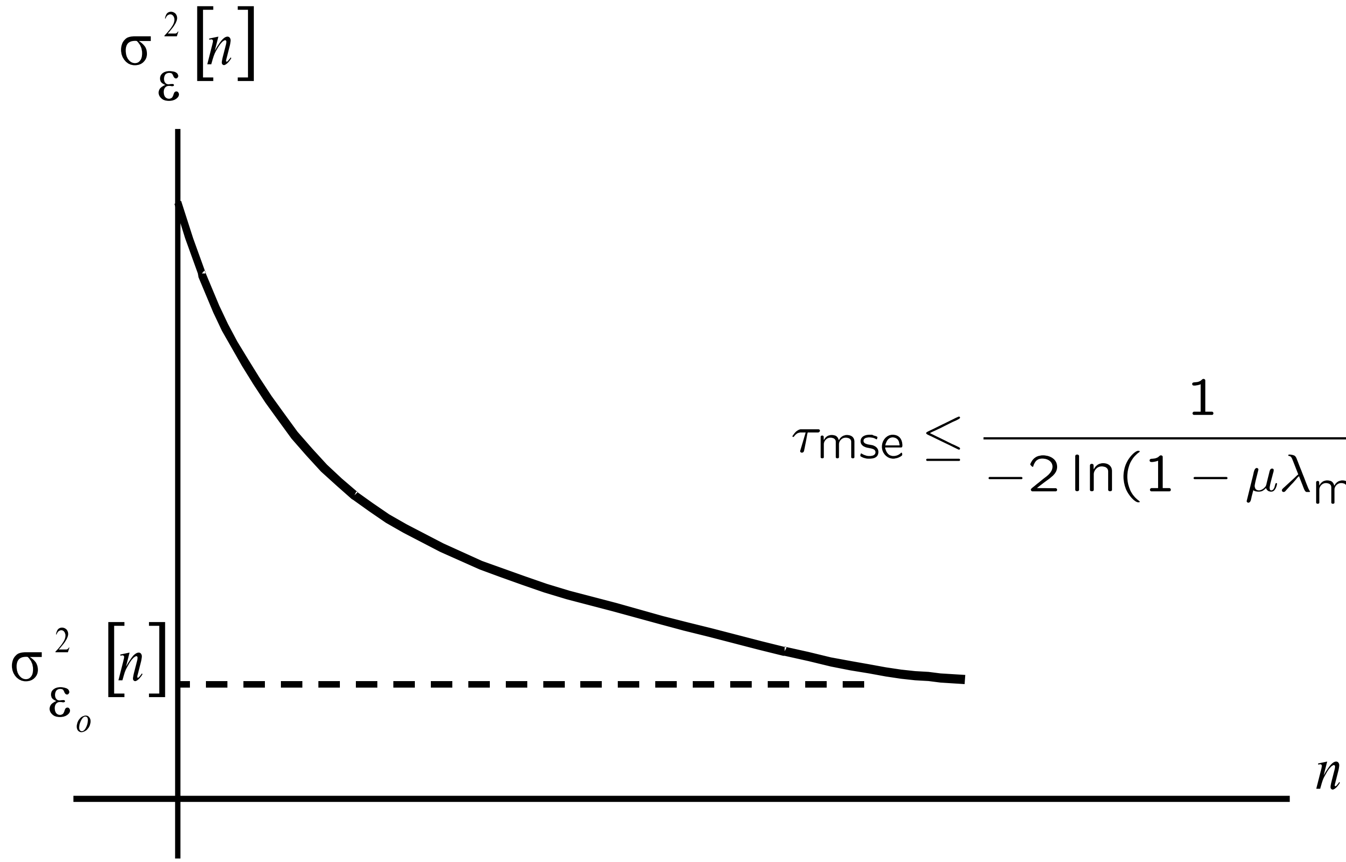
$\lambda_{\mathsf{max}}/\lambda_{\mathsf{min}}$ is the *condition number* χ of the correlation matrix.

CONVERGENCE OF MEAN-SQUARE ERROR



$$\sigma_\varepsilon^2[n] = \sigma_{\varepsilon_\mathrm{o}}^2 + \sum_{i=1}^{P} \lambda_i (1 - \mu\lambda_i)^{2n} |v_i[0]|^2$$

LEARNING CURVE (MEAN-SQUARE ERROR)



$$\tau_{\text{mse}} \leq \frac{1}{-2\ln(1-\mu\lambda_{\text{min}})} \leq \frac{1}{2\mu\lambda_{\text{min}}}$$

LEAST MEAN SQUARES (LMS) ALGORITHM

EQUATION OF STEEPEST DESCENT

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu \mathcal{E}\{\varepsilon[n]\tilde{\boldsymbol{x}}^*[n]\}$$

LMS ALGORITHM

$$\boxed{\begin{array}{rcll} \varepsilon[n] & = & d[n] - \mathbf{w}^T[n]\tilde{\boldsymbol{x}}[n] & \text{(a)} \\ \mathbf{w}[n+1] & = & \mathbf{w}[n] + \mu\,\varepsilon[n]\tilde{\boldsymbol{x}}^*[n] & \text{(b)} \end{array}}$$

$$0 < \mu < \frac{2}{\text{tr } \mathbf{R}_{\boldsymbol{x}}} = \frac{2}{\text{tap input power}} = \frac{2}{PR_x[0]}$$

LMS CONVERGENCE

CONVERGENCE OF THE MEAN

$$\lim_{n\to\infty} \mathcal{E}\{\varepsilon[n]\} = \varepsilon_\infty \quad \text{and} \quad \lim_{n\to\infty} \mathcal{E}\{\mathbf{w}[n]\} = \mathbf{w}_\infty$$

CONVERGENCE IN MEAN-SQUARE

$$\lim_{n\to\infty} \mathcal{E}\left\{|\varepsilon[n]|^2\right\} = \sigma_\varepsilon^2$$

MISADJUSTMENT

$$\mathcal{M} \stackrel{\text{def}}{=} \frac{\sigma_\varepsilon^2 - \sigma_{\varepsilon_\text{o}}^2}{\sigma_{\varepsilon_\text{o}}^2} = \frac{\sigma_\varepsilon^2}{\sigma_{\varepsilon_\text{o}}^2} - 1$$

DISCUSSION OF LMS

- Most widely used adaptive algorithm

- Low computational requirements: $\mathcal{O}(P)$

- Nonlinear and time-varying algorithm;
  extremely difficult to analyze

- Steepest descent method provides guidelines for LMS

LMS VARIATIONS

- Leaky LMS
- Normalized LMS
- Sign algorithms
- Quasi-Newton methods

LEAKY LMS

$$\boxed{\mathbf{w}[n+1] = (1-\mu\alpha)\mathbf{w}[n] + \mu\,\varepsilon[n]\tilde{\boldsymbol{x}}^*[n]}$$

α is a small positive number; $\quad 0 < \mu < \dfrac{2}{\alpha + \lambda_{\mathsf{max}}}$

- mitigates nonconverging modes
- introduces additional error bias

NORMALIZED LMS

$$\boxed{\mathbf{w}[n+1] = \mathbf{w}[n] + \frac{\mu'}{\epsilon + \|\tilde{\boldsymbol{x}}[n]\|^2}\,\varepsilon[n]\tilde{\boldsymbol{x}}^*[n]}$$

ϵ is a small positive number; $\quad 0 < \mu' < 2$

- uses normalized step size parameter
- eliminates need to estimate upper bound on μ

SIGN ALGORITHMS

$$\boxed{\mathbf{w}[n+1] = \mathbf{w}[n] + \mu \, \mathsf{sgn}\,\{\varepsilon[n]\}\, \tilde{\boldsymbol{x}}[n]}$$

$$\boxed{\mathbf{w}[n+1] = \mathbf{w}[n] + \mu \, \mathsf{sgn}\,\{\varepsilon[n]\}\, \mathsf{sgn}\,\{\tilde{\boldsymbol{x}}[n]\}}$$

- computationally simplest

- may have slow convergence and/or stability problems

QUASI-NEWTON ALGORITHMS

$$\boxed{\mathbf{w}[n+1] = \mathbf{w}[n] + \mu\,\varepsilon[n]\hat{\mathbf{R}}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}^*[n]}$$

$\hat{\mathbf{R}}_{\boldsymbol{x}}^{-1}$ is an approximation to the true inverse

- speeds convergence by use of second order terms
- whitens the input
- increased computation: $\mathcal{O}(P^2)$

LMS FOR LATTICES (LINEAR PREDICTION)

- Minimize combined criterion

$$J_p = \mathcal{E}\left\{|\varepsilon_p[n]|^2 + |\varepsilon_p^b[n]|^2\right\}$$

- Direction of steepest descent is

$$-\nabla_{\gamma_p^*} J_p = \mathcal{E}\left\{\varepsilon_{p-1}^b[n-1]\varepsilon_p^*[n] + \varepsilon_p^b[n]\varepsilon_{p-1}^*[n]\right\}$$

- Requires double recursion: in *order* as well as time

GRADIENT ADAPTIVE LATTICE (GAL) ALGORITHM

- Minimizes weighted sum of errors ($0 < \beta < 1$)

$$\mathcal{S}_p^{fb}[n] = \beta \mathcal{S}_p^{fb}[n-1] + (1-\beta)\left[|\varepsilon_p[n]|^2 + |\varepsilon_p^b[n-1]|^2\right]$$

1. Start with $\varepsilon_p[-1] = \varepsilon_p^b[-1] = 0$, $\gamma_p[-1] = 0$, and $\mathcal{S}_{p-1}^{fb}[-1] = \epsilon$ (a small number) for all orders $p \leq P$.

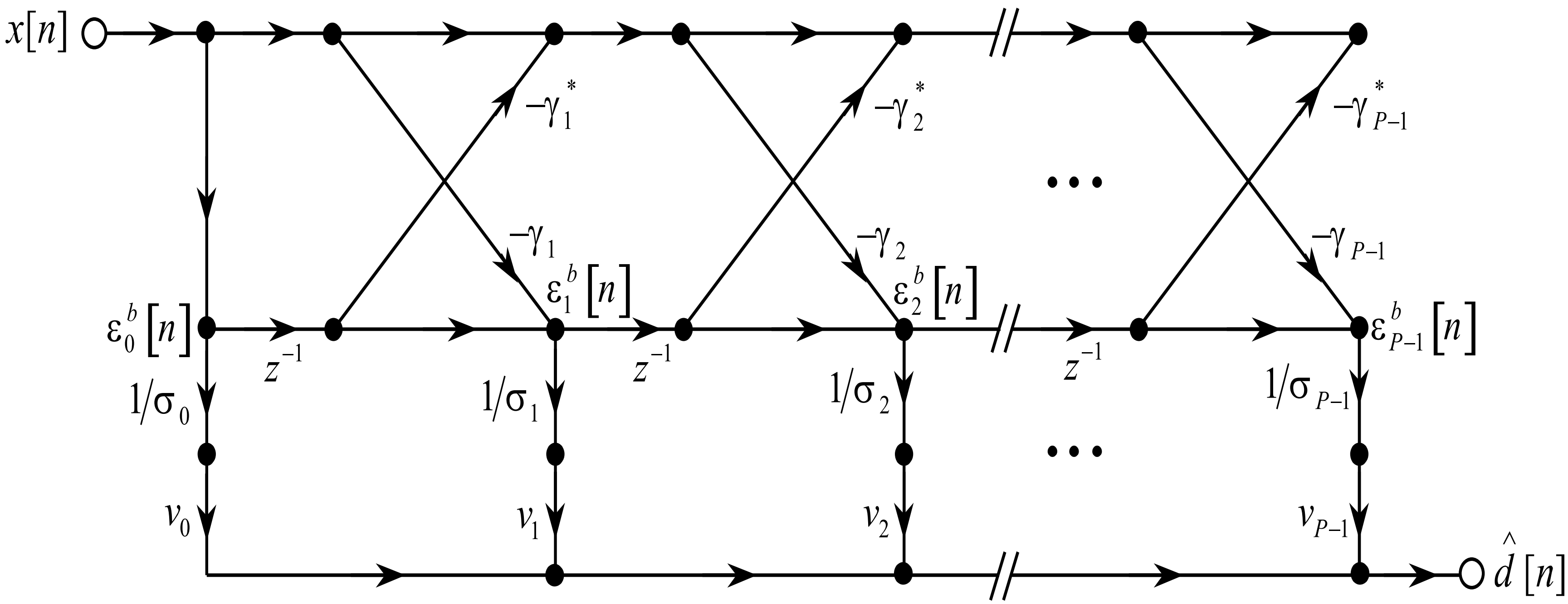
2. For $n = 0, 1, 2, \ldots$

   (a) Set $\varepsilon_0[n] = \varepsilon_0^b[n] = x[n]$.

(b) For $p = 1, 2, \ldots, P$ compute:

$$\begin{aligned}
\varepsilon_p[n] &= \varepsilon_{p-1}[n] - \gamma_p^* \varepsilon_{p-1}^b[n-1] \\
\varepsilon_p^b[n] &= \varepsilon_{p-1}^b[n-1] - \gamma_p \varepsilon_{p-1}[n] \\
\mathcal{S}_p^{fb}[n] &= \beta \mathcal{S}_p^{fb}[n-1] + (1-\beta)\left[|\varepsilon_p[n]|^2 + |\varepsilon_p^b[n-1]|^2 \right] \\
\mu_p[n] &= \frac{\tilde{\mu}}{\mathcal{S}_{p-1}^{fb}[n]} \qquad\qquad 0 < \tilde{\mu} < 0.1 \\
\gamma_p[n] &= \gamma_p[n-1] + \mu_p[n]\left[\varepsilon_{p-1}^b[n-1]\varepsilon_p^*[n] + \varepsilon_p^b[n]\varepsilon_{p-1}^*[n] \right]
\end{aligned}$$

"JOINT PROCESS" ESTIMATION



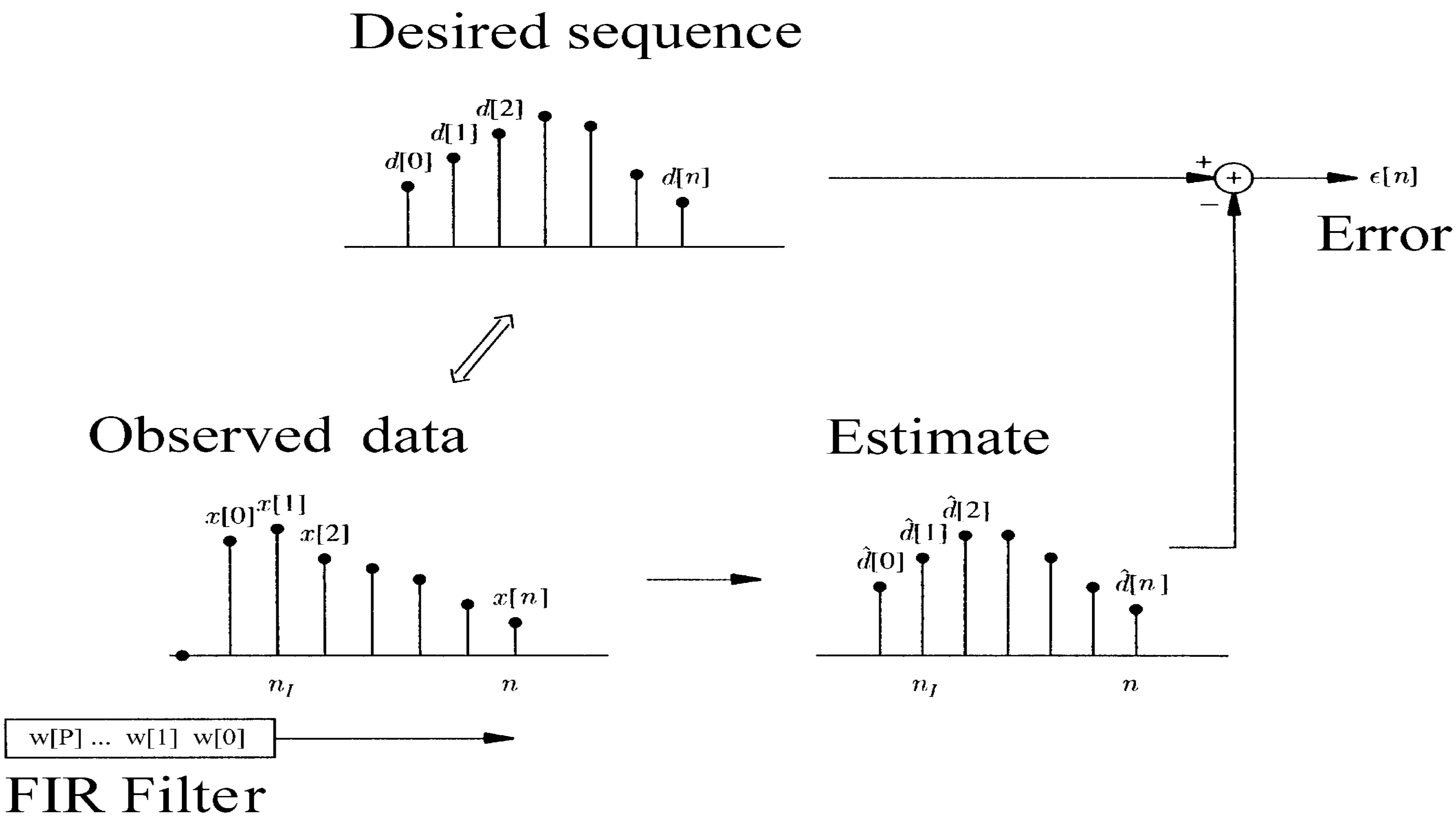
- Add step: $v_p[n+1] = v_p[n] + \dfrac{\mu'}{\mathcal{S}_p^{fb}[n]}\varepsilon[n]\varepsilon_p^{b*}[n] \qquad p = 0, 1, \ldots, P$

RECURSIVE LEAST SQUARES FILTERING

- Weighted least squares

- The RLS algorithm

LEAST SQUARES FILTERING (REVIEW)



WEIGHTED LEAST SQUARES

Minimize:

$$\mathcal{S}_\beta[n] \stackrel{\text{def}}{=} \sum_{i=n_I}^{n} \beta^{n-i} |\epsilon[i]|^2 \,; \qquad 0 < \beta < 1$$

$$= \boldsymbol{\epsilon}[n]^{*T} \mathbf{B}[n] \boldsymbol{\epsilon}[n]$$

where

$$\boldsymbol{\epsilon}[n] = \begin{bmatrix} \epsilon[n_I] \\ \epsilon[n_I+1] \\ \vdots \\ \\ \vdots \\ \epsilon[n] \end{bmatrix} ; \qquad \mathbf{B}[n] = \begin{bmatrix} \beta^{n-n_I} & 0 & \cdots & 0 \\ 0 & \ddots & & \\ \vdots & & \beta & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

WEIGHTED LEAST SQUARES SOLUTION

$$\boxed{\mathbf{R}[n]\mathbf{w}[n] = \mathbf{r}[n]} \qquad \Longrightarrow \qquad \mathbf{w}[n] = \mathbf{R}^{-1}[n]\mathbf{r}[n]$$

$$\mathbf{R}[n] = \mathbf{X}^{*T}[n]\mathbf{B}[n]\mathbf{X}[n]\,; \qquad \mathbf{r}[n] = \mathbf{X}^{*T}[n]\mathbf{B}[n]\mathbf{d}[n]$$

$$\mathbf{X}[n]=\begin{bmatrix} \mathsf{x}[n_I] & \mathsf{x}[n_I-1] & \cdots & \mathsf{x}[n_I-P+1] \\ \mathsf{x}[n_I+1] & \mathsf{x}[n_I] & \cdots & \mathsf{x}[n_I-P+2] \\ \vdots & \vdots & & \vdots \\ \\ \vdots & \vdots & & \vdots \\ \mathsf{x}[n] & \mathsf{x}[n-1] & \cdots & \mathsf{x}[n-P+1] \end{bmatrix} \qquad \mathbf{d}[n]=\begin{bmatrix} \mathsf{d}[n_I] \\ \mathsf{d}[n_I+1] \\ \vdots \\ \\ \vdots \\ \mathsf{d}[n] \end{bmatrix}$$

RECURSION FOR FILTER COEFFICIENTS

RECURSIVE EXPRESSIONS FOR VARIABLES

$$\mathbf{X}[n] = \begin{bmatrix} \mathbf{X}[n{-}1] \\ \tilde{\mathbf{x}}^T[n] \end{bmatrix} \qquad \mathbf{d}[n] = \begin{bmatrix} \mathbf{d}[n{-}1] \\ \mathsf{d}[n] \end{bmatrix} \qquad \mathbf{B}[n] = \begin{bmatrix} \beta\mathbf{B}[n{-}1] & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\tilde{\mathbf{x}}^T[n] = \begin{bmatrix} \mathsf{x}[n{-}P{+}1] & \cdots & \mathsf{x}[n{-}1] & \mathsf{x}[n] \end{bmatrix}$$

RECURSION (cont'd.)

TERMS IN WIENER-HOPF EQUATION

$$
\begin{aligned}
\mathbf{R}[n] &= \mathbf{X}^{*T}[n]\mathbf{B}[n]\mathbf{X}[n] \\
&= \left[\begin{array}{cc} \mathbf{X}^{*T}[n-1] & \tilde{\mathbf{x}}^{*}[n] \end{array} \right] \left[\begin{array}{cc} \beta\mathbf{B}[n-1] & \mathbf{0} \\ \mathbf{0} & 1 \end{array} \right] \left[\begin{array}{c} \mathbf{X}[n-1] \\ \tilde{\mathbf{x}}^{T}[n] \end{array} \right] \\
&= \beta\mathbf{R}[n-1] + \tilde{\mathbf{x}}^{*}[n]\tilde{\mathbf{x}}^{T}[n]
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{r}[n] &= \mathbf{X}^{*T}[n]\mathbf{B}[n]\mathbf{d}[n] \\
&= \left[\begin{array}{cc} \mathbf{X}^{*T}[n-1] & \tilde{\mathbf{x}}^{*}[n] \end{array} \right] \left[\begin{array}{cc} \beta\mathbf{B}[n-1] & \mathbf{0} \\ \mathbf{0} & 1 \end{array} \right] \left[\begin{array}{c} \mathbf{d}[n-1] \\ \mathsf{d}[n] \end{array} \right] \\
&= \beta\mathbf{r}[n-1] + \tilde{\mathbf{x}}^{*}[n]\mathsf{d}[n]
\end{aligned}
$$

RECURSION (cont'd.)

SHERMAN-MORRISON IDENTITY

$$(\mathbf{A}+\mathbf{b}\mathbf{c}^{*T})^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{b}\,\mathbf{c}^{*T}\mathbf{A}^{-1}}{1+\mathbf{c}^{*T}\mathbf{A}^{-1}\mathbf{b}}$$

INVERSE CORRELATION MATRIX

$$\begin{aligned}(\mathbf{R}[n])^{-1} &= (\beta\mathbf{R}[n\text{–}1])^{-1} - \frac{(\beta\mathbf{R}[n\text{–}1])^{-1}\tilde{\mathbf{x}}^{*}[n]\tilde{\mathbf{x}}^{T}[n](\beta\mathbf{R}[n\text{–}1])^{-1}}{1+\tilde{\mathbf{x}}^{T}[n](\beta\mathbf{R}[n\text{–}1])^{-1}\tilde{\mathbf{x}}^{*}[n]} \\ &= \left(\mathbf{I} - \frac{(\beta\mathbf{R}[n\text{–}1])^{-1}\tilde{\mathbf{x}}^{*}[n]}{1+\tilde{\mathbf{x}}^{T}[n](\beta\mathbf{R}[n\text{–}1])^{-1}\tilde{\mathbf{x}}^{*}[n]}\cdot\tilde{\mathbf{x}}^{T}[n]\right)(\beta\mathbf{R}[n\text{–}1])^{-1}\end{aligned}$$

RECURSION (cont'd.)

INVERSE CORRELATION SIMPLIFIED

$$\mathbf{R}^{-1}[n] = \beta^{-1}(\mathbf{I} - \mathbf{k}[n]\tilde{\mathbf{x}}^T[n])\mathbf{R}^{-1}[n-1]$$

where

$$\mathbf{k}[n] = \frac{\beta^{-1}\mathbf{R}^{-1}[n-1]\tilde{\mathbf{x}}^*[n]}{1 + \beta^{-1}\tilde{\mathbf{x}}^T[n]\mathbf{R}^{-1}[n-1]\tilde{\mathbf{x}}^*[n]}$$

It can further be shown that

$$\mathbf{k}[n] = \mathbf{R}^{-1}[n]\tilde{\mathbf{x}}^*[n]$$

RECURSION (cont'd.)

FILTER WEIGHT VECTOR

Substitute previous equations:

$$\begin{aligned}
\mathbf{w}[n] &= \mathbf{R}^{-1}[n]\mathbf{r}[n] = \mathbf{R}^{-1}[n](\beta\mathbf{r}[n-1] + \tilde{\mathbf{x}}^*[n]\mathsf{d}[n]) \\
&= \beta\mathbf{R}^{-1}[n]\mathbf{r}[n-1] + \mathbf{k}[n]\mathsf{d}[n] \\
&= \beta\left(\beta^{-1}(\mathbf{I} - \mathbf{k}[n]\tilde{\mathbf{x}}^T[n])\mathbf{R}^{-1}[n-1]\right)\mathbf{r}[n-1] + \mathbf{k}[n]\mathsf{d}[n] \\
&= (\mathbf{I} - \mathbf{k}[n]\tilde{\mathbf{x}}^T[n])\mathbf{w}[n-1] + \mathbf{k}[n]\mathsf{d}[n]
\end{aligned}$$

to obtain:

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \mathbf{k}[n]\underbrace{\left(\mathsf{d}[n] - \mathbf{w}^T[n-1]\tilde{\mathbf{x}}[n]\right)}_{\mathsf{e}[n]}$$

RLS ALGORITHM

ESSENTIAL STEPS

compute gain: $$\mathbf{k}[n] = \frac{\beta^{-1}\mathbf{R}^{-1}[n-1]\tilde{\mathbf{x}}^*[n]}{1+\beta^{-1}\tilde{\mathbf{x}}^T[n]\mathbf{R}^{-1}[n-1]\tilde{\mathbf{x}}^*[n]}$$

compute prior error: $$\mathsf{e}[n] = \mathsf{d}[n] - \mathbf{w}^T[n-1]\tilde{\mathbf{x}}[n]$$

update weights: $$\mathbf{w}[n] = \mathbf{w}[n-1] + \mathbf{k}[n]\mathsf{e}[n]$$

update inverse matrix: $$\mathbf{R}^{-1}[n] = \beta^{-1}(\mathbf{I} - \mathbf{k}[n]\tilde{\mathbf{x}}^T[n])\mathbf{R}^{-1}[n-1]$$

FURTHER DISCUSSION OF RLS

- Additional parameters and interpretation
- Weighted sum of squared errors
- Computational requirements
- Demonstration of performance

RLS ALGORITHM (EXPANDED)

$$\begin{aligned}
\mathbf{k}'[n] &= \beta^{-1}\mathbf{R}^{-1}[n-1]\tilde{\mathbf{x}}^*[n] && \text{(a)} \\
\kappa[n] &= 1/(1+\mathbf{k}'^T[n]\tilde{\mathbf{x}}[n]) && \text{(b)} \\
\mathbf{k}[n] &= \kappa[n]\mathbf{k}'[n] && \text{(c)} \\
\mathsf{e}[n] &= \mathsf{d}[n] - \mathbf{w}^T[n-1]\tilde{\mathbf{x}}[n] && \text{(d)} \\
\mathbf{w}[n] &= \mathbf{w}[n-1] + \mathbf{k}[n]\mathsf{e}[n] && \text{(e)} \\
\mathbf{R}^{-1}[n] &= \beta^{-1}\mathbf{R}^{-1}[n-1] - \mathbf{k}[n](\mathbf{k}'[n])^{*T} && \text{(f)}
\end{aligned}$$

typical initialization: $\mathbf{R}^{-1}[0] = \frac{1}{\delta}\mathbf{I}\,, \qquad \mathbf{w}[0] = \mathbf{0}$

ERROR TERMS IN RLS

PRIOR ERROR

$$\mathsf{e}[n] = \mathsf{d}[n] - \mathbf{w}^T[n-1]\tilde{\mathbf{x}}[n]$$

POSTERIOR ERROR

$$\epsilon[n] = \mathsf{d}[n] - \mathbf{w}^T[n]\tilde{\mathbf{x}}[n]$$

RELATION

$$\epsilon[n] = \kappa[n]\mathsf{e}[n] \qquad 0 < \kappa[n] < 1$$

$\kappa[n]$ is called the *conversion factor*.

GAIN TERMS IN RLS

WEIGHT UPDATE EQUATION (PRIOR FORM)

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \mathbf{k}[n]\mathsf{e}[n]$$

WEIGHT UPDATE EQUATION (POSTERIOR FORM)

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \mathbf{k}'[n]\epsilon[n]$$

RELATION BETWEEN GAIN TERMS

$$\mathbf{k}[n] = \kappa[n]\,\mathbf{k}'[n] \qquad 0 < \kappa[n] < 1$$

WEIGHTED SUM OF SQUARES

UPDATE FORMULA

$$\mathcal{S}_\beta[n] = \beta \mathcal{S}_\beta[n-1] + \epsilon^*[n]\mathsf{e}[n]$$

ALTERNATIVE FORMS

$$\begin{aligned} \mathcal{S}_\beta[n] &= \beta \mathcal{S}_\beta[n-1] + \kappa[n]|\mathsf{e}[n]|^2 \quad \text{(a)} \\ \mathcal{S}_\beta[n] &= \beta \mathcal{S}_\beta[n-1] + |\epsilon[n]|^2/\kappa[n] \quad \text{(b)} \end{aligned}$$

RLS ALGORITHM IN MATLAB

MAIN LOOP

```
for n=P:length(x)
xb=x(n-P+1:n);  g=Rm1*xb';
gamma=1/(beta + xb*g);
k=gamma*g;
epr(n) = d(n) - xb*W(:,n-1);
W(:,n) = W(:,n-1) + k*epr(n);
Rm1 = (Rm1 - k*g')/beta;
epo(n) = beta*gamma*epr(n);
S(n) = beta*S(n-1) + epo(n)*conj(epr(n));
end
```